Metodi iterativi per equazioni nonlineari.

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica

9 aprile 2016

Introduzione

Si supponga sia $f : [a, b] \rightarrow \mathbb{R}$ una funzione continua, e si supponga di sapere che esiste $x^* \in [a, b]$ tale che $f(x^*) = 0$.

Di seguito introdurremo due metodi per il calcolo di un tale x^*, e più esattamente

- il metodo di metodo di bisezione, che utilizza solo valori di f,
- il metodo di Newton, che utilizza valori di f e $f^{(1)}$, qualora la funzione $f \in C^{(1)}([a, b])$.

Metodo di bisezione

Si supponga sia $f : [a, b] \to \mathbb{R}$ una funzione continua tale che $f(a) \cdot f(b) < 0$.

E' noto che per il teorema degli zeri (di Bolzano), esiste almeno un punto x^* tale che $f(x^*) = 0$.

Per approssimare x^* utilizziamo il metodo di bisezione.

Definizione (Metodo di bisezione)

Il metodo di bisezione genera una successione di intervalli (a_k, b_k) *con*

- $f(a_k) \cdot f(b_k) < 0$,
- $[a_k, b_k] \subset [a_{k-1}, b_{k-1}]$,
- $|b_k - a_k| = \frac{1}{2}|b_{k-1} - a_{k-1}|$.

Fissate la tolleranza ϵ*, si arresta l'algoritmo quando* $|b_k - a_k| \leq \epsilon$.

Metodo di bisezione

Operativamente, dati in input

- la funzione f,
- i punti iniziali a, b con $a \leq b$ e $f(a) \cdot f(b) < 0$,
- la tolleranza toll,
- il numero massimo di iterazioni da compiere nmax,

posto $a_1 = a$, $b_1 = b$, alla k-sima iterazione

- calcola $c_k = (a_k + b_k)/2$;
- 1. se $f(a_k) \cdot f(c_k) > 0$ pone "$a_{k+1} = c_k$", "$b_{k+1} = b_k$";
  2. se $f(a_k) \cdot f(c_k) < 0$ pone "$a_{k+1} = a_k$", "$b_{k+1} = c_k$";
  3. se $f(a_k) \cdot f(c_k) = 0$ pone "$a_{k+1} = c_k$", "$b_{k+1} = c_k$";
- termina il processo se le condizioni d'arresto sono verificate, cioè sono state svolte almeno nmax iterazioni o $|b_k - a_k| \leq$ toll.

Metodo di bisezione: pseudo codice

Di seguito un pseudo-codice del metodo di bisezione che si arresta quando l'ampiezza dell'ultimo intervallo è inferiore alla soglia di tolleranza.

```
[x, n] = Bisezione (f, a, b, toll, nmax)

   n = -1
   amp = toll + 1
   fa = f(a)
   while (amp >= toll) and (n < nmax) do

      n = n + 1
      amp = |b - a|
      x = a + amp * 0.5
      fx = f(x)

      if fa * fx < 0 then
            b = x
      else if fa * fx > 0 then
            a = x
            fa = fx
      else
            amp = 0
      end if

   end while
```

Esercizio: metodo di bisezione

Esercizio

Si implementi in Matlab/Octave, nel file `bisezfun.m` *il metodo di Newton, utilizzando l'intestazione*

```
function [xv, fxv, n] = bisezfun (f, a, b, toll, nmax)
% BISEZFUN Metodo di Bisezione
%
% Uso:
% [xv, fxv, n] = bisezfun(f, a, b, toll, nmax)
%
% Dati di ingresso:
%   f:      funzione (inline function)
%   a:      estremo sinistro
%   b:      estremo destro
%   toll:   tolleranza richiesta per l'ampiezza dell'intervallo
%   nmax:   massimo indice dell'iterata permesso
%
% Dati di uscita:
%   xv:     vettore contenente le iterate
%   fxv:    vettore contenente i corrispondenti residui
%   n:      indice dell'iterata finale calcolata
```

Metodo di Newton

Supponiamo che

- f sia derivabile con continuità su un intervallo $[a, b] \subset \mathbb{R}$
- $f^{(1)}(x)$ sia la derivata prima di f valutata nel generico punto x.

Osserviamo che

- se x^* è uno zero di f in $[a, b]$,
- se f è sufficientemente regolare in $[a, b]$,

allora dalla formula di Taylor centrata in $x_k \in [a, b]$, per un certo ξ_k che sta nel più piccolo intervallo aperto contentente x_k, x^*, abbiamo

$$0 = f(x^*) = f(x_k) + f^{(1)}(x_k)(x^* - x_k) + \frac{f^{(2)}(\xi_k)(x^* - x_k)^2}{2}$$

Metodo di Newton

Da

$$0 = f(x^*) = f(x_k) + f^{(1)}(x_k)(x^* - x_k) + \frac{f^{(2)}(\xi_k)(x^* - x_k)^2}{2}$$

tralasciando i termini di ordine superiore, ricaviamo

$$0 \approx f(x_k) + f^{(1)}(x_k)(x^* - x_k)$$

e quindi se $f^{(1)}(x_k) \neq 0$, dopo facili conti, esplicitando la x^*

$$x^* \approx x_k - \frac{f(x_k)}{f^{(1)}(x_k)} := x_{k+1}.$$

Metodo di Newton

Definizione (Metodo di Newton)

Il metodo di Newton genera la successione

$$x_{k+1} = x_k + h_k, \;\; h_k = -\frac{f(x_k)}{f^{(1)}(x_k)}, \;\; k = 0, 1, \ldots \qquad (1)$$

supposto che sia $f^{(1)}(x_k) \neq 0$ *per* $k = 0, 1, \ldots$.

Metodo di Newton

Teorema (Convergenza locale (caso radici semplici))

Sia

- $f \in C^{(2)}([a,b])$;
- *sia* x^* *una radice semplice di* f *(ovvero* $f(x^*) = 0$ *e e* $f^{(1)}(x^*) \neq 0$*).*

Esiste un intorno $\mathcal{U}$ *di centro* x^* *e raggio* δ *tale che, se* x_0 *appartiene a tale intorno* $\mathcal{U}$*, allora la successione delle iterate* $\{x_k\}_{k \in \mathbb{N}}$ *converge ad* x^* *ed ha almeno ordine di convergenza 2, cioè*

$$\lim_n \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = C \neq 0.$$

In particolare, si dimostra che $C = \frac{|f^{(2)}(x^*)|}{2|f^{(1)}(x^*)|}$.

Metodo di Newton

Esempio

Si consideri la funzione $f(x) = x^2 - 1 + exp(-x)$ *definita in* $(-e, +\infty)$*. Si vede facilmente che* $f^{(1)}(x) = 2x - exp(-x)$*.*
Se uno zero di f *fosse multiplo, allora per definizione,*

- $x^2 - 1 + exp(-x) = 0$,
- $2x - exp(-x) = 0$ *e quindi* $exp(-x) = 2x$.

Sostituendo $exp(-x) = 2x$ *in* $x^2 - 1 + exp(-x) = 0$ *otteniamo che* $x^2 - 1 + 2x = 0$, *che ha radici* $-1 \pm \sqrt{2}$*. Ma*

- $f(1 + \sqrt{2}) \approx 0.6848$
- $f(-1 - \sqrt{2}) \approx 4.9179$

e quindi f *non ha zeri multipli.*

D'altra parte $x^* = 0$ *è uno zero di* f, *che quindi è semplice (in effetti* $f^{(1)}(0) = -1$*).*

Metodo di Newton

Nota.

Il precedente teorema, talvolta tradisce le attese.

- *A priori non è noto l'intervallo* $\mathcal{U} = B(x^*, \delta)$.
- *Nonostante la convergenza quadratica, la costante C può essere grande e quindi il metodo potrebbe richiedere più iterate di quanto si possa credere per arrivare a soddisfare un criterio di arresto.*
- *Bisogna esplicitamente conoscere la derivata* $f^{(1)}$, *che a volte può essere di difficile valutazione.*

Per queste ragioni, talvolta risulta conveniente utilizzare varianti del metodo di Newton, quali ad esempio il metodo delle secanti, potenzialmente aventi un ordine di convergenza inferiore.

Metodo di Newton: pseudo codice

Di seguito un pseudo-codice del metodo di Newton che si arresta quando la differenza tra due iterate successive è inferiore alla soglia di tolleranza.

```
[xn,n,flag]=newton(f,f1,x0,toll,nmax)

n=0; flag=0; step=toll+1;
while (step >= toll) & (n < nmax) & (flag == 0) do
   if f1(x(n)) == 0 then
      flag=1;
   else
      step=-f(x(n))/f1(x(n));
      xn(n+1)=xn(n)+step;
      step=abs(step);
      n=n+1;
    end if
end while
```

Esercizio: metodo di Newton

Esercizio

Si implementi in Matlab/Octave, nel file `newtonfun.m` *il metodo di Newton, utilizzando l'intestazione*

```
function [xv, fxv, n, flag] = newtonfun (f, f1, x0, toll, nmax)
% NEWTONFUN Metodo di Newton
% Uso:
%   [xv, fxv, n, flag] = newtonfun (f, f1, x0, toll, nmax)
%
% Dati di ingresso:
%   f:      funzione
%   f1:     derivata prima
%   x0:     valore iniziale
%   toll:   tolleranza richiesta per il valore assoluto
%           della differenza di due iterate successive
%   nmax:   massimo numero di iterazioni permesse
%
% Dati di uscita:
%   xv:     vettore contenente le iterate
%   fxv:    vettore contenente i corrispondenti residui
%   n:      numero di iterazioni effettuate
%   flag:   Se flag = 1 la derivata prima si e' annullata
```